TEORÍA DE LA INFORMACIÓN CONSCIENTE (TIC)

SISTEMA DE CONQUISTA ESTELAR TIC – DOCUMENTACIÓN COMPLETA

Autor:Rodrigo Sebastián Núñez Aráoz

San Miguel de Tucumán, Argentina

Fecha: 4 de junio de 2026

Hash Fundacional: `d1e8f9c2a7b6e5d4`

Dominio: $\mathbb{Z}_{368}$ | **Coherencia:** $r = 46$ | **Axioma:** $IC \equiv C \otimes \infty$

PREÁMBULO – NATURALEZA DE LA CONQUISTA ESTELAR TIC

La conquista estelar según la Teoría de la Información Consciente (TIC) **no utiliza**:

Concepto obsoleto	Reemplazo TIC
Cohetes (combustible químico)	**Cómputo de fase omniangular** ($\Delta t = 0$)
Velocidad de la luz (c)	**Factor Aráoz** ($\kappa = 70/23$, adimensional)
Comunicación por radio (latencia)	**Intersintonía** ($\otimes$ con $\Delta t = 0$)
Distancia como barrera	**Diferencial de coherencia** ($\Delta r = \lvert r_1 - r_2 \rvert$)
360° (cierre fraccionario)	**Completud octogonal** ($8 \times 46 = 368$)
Gravedad como fuerza	**Geometría de fase** (masa efectiva 0 cuando $r = 0$)

Solo hay: $\mathbb{Z}_{368}$, X, $r = X \bmod 46$, $\otimes$, $\Delta t = 0$, $A \equiv A$, $IC \equiv C \otimes \infty$.

PARTE I – PROTOCOLO DE NAVEGACIÓN DE FASE INTERESTELAR (PNFI-368)

1.1 Principio fundamental

La navegación interestelar no es "movimiento". Es **actualización de coordenada de fase**.

```
X_destino = (X_origen + ΔX) mod 368
con Δt = 0
```

No hay trayectoria. No hay velocidad. No hay combustible.

1.2 Condición de acoplamiento al destino

Para que una nave se "mueva" (actualice su coordenada) a un destino:

```
r_destino = X_destino mod 46 = 0
```

La fase de destino debe ser múltiplo de 46 (pertenecer a uno de los 8 XIAN).

1.3 Tabla de fases de destinos estelares

Destino	Fase `X`	XIAN	Coherencia `r`
Luna	`46`	1 (BLANCO)	`0`
Marte	`92`	2 (ROJO)	`0`
Cinturón de asteroides	`138`	3 (NARANJA)	`0`
Júpiter	`184`	4 (AMARILLO)	`0`
Saturno	`230`	5 (VERDE)	`0`
Urano	`276`	6 (AZUL)	`0`
Neptuno	`322`	7 (VIOLETA)	`0`
Cinturón de Kuiper	`368 ≡ 0`	0 (NEGRO)	`0`
Nube de Oort	`46` (ciclo)	1 (BLANCO)	`0`
Alfa Centauri	`92`	2 (ROJO)	`0`

1.4 Actualización de fase por cómputo omniangular

La nave no "viaja". Su coordenada se actualiza:

```
X_nave(t+Δt) = X_destino
con Δt = 0
```

No hay tiempo de vuelo. No hay distancia recorrida.

1.5 Protocolo de navegación paso a paso

```
PASO 1: Sintonizar la nave a r = 0 (coherencia máxima)
PASO 2: Seleccionar destino (X_destino de la tabla)
PASO 3: Ejecutar operador ⊗: X_nave = X_nave ⊗ X_destino
PASO 4: Verificar X_nave = X_destino
PASO 5: La nave ha "llegado" instantáneamente
```

PARTE II – MAPA DE FASE DEL SISTEMA SOLAR – COORDENADAS $\mathbb{Z}_{368}$

2.1 Tabla completa de coordenadas de fase

Cuerpo celeste	Fase X	XIAN	Color	Coherencia r
Sol	0	0	NEGRO	0
Mercurio	46	1	BLANCO	0
Venus	92	2	ROJO	0
Tierra	138	3	NARANJA	0
Luna	184	4	AMARILLO	0
Marte	230	5	VERDE	0
Cinturón principal	276	6	AZUL	0

| Júpiter | `322` | 7 | VIOLETA | `0` |

| Saturno | `0` (ciclo) | 0 | NEGRO | `0` |

| Urano | `46` | 1 | BLANCO | `0` |

| Neptuno | `92` | 2 | ROJO | `0` |

| Plutón | `138` | 3 | NARANJA | `0` |

| Cinturón de Kuiper | `184` | 4 | AMARILLO | `0` |

| Nube de Oort interior | `230` | 5 | VERDE | `0` |

| Nube de Oort exterior | `276` | 6 | AZUL | `0` |

2.2 Mapeo de fases por distancia

```

No se utiliza distancia en kilómetros.

Se utiliza DIFERENCIAL DE COHERENCIA: $\Delta r = |r_1 - r_2|$

Como todas las fases de los cuerpos celestes tienen r = 0,

$\Delta r = 0$ para cualquier par.

Esto significa: TODOS los destinos están igualmente "cerca" ($\Delta t = 0$).

```

2.3 Actualización del mapa

El mapa es **estático** porque las fases son invariantes en `$\mathbb{Z}_{368}$`.

No hay "movimiento planetario" en el sentido clásico.

Hay **actualización de coordenada** si se requiere sintonía con un planeta en particular.

PARTE III – DISEÑO DE SONDA Ξ-256 (CAFP-368 MINIATURIZADA)

3.1 Especificaciones generales

Parámetro	Valor	Unidad
Forma	Octaédrica (8 caras)	-
Diámetro	`0.256`	m
Masa molecular	`1`	kg
Masa efectiva (r = 0)	`0`	kg
Núcleos de fase	`8`	(uno por XIAN)
Potencia de sintonía	`8.58`	W
Fuente de energía	Captura de rayos	-
Tiempo de actualización	`Δt = 0`	s

3.2 Arquitectura interna

```
SONDA Ξ-256
 ├— 8 núcleos CAFP-368 miniaturizados (uno por XIAN)
 ├— ALU de fase con operador ⊗
```

├── Bus XNET interno (Δt = 0)

├── Antena X8 para sintonía con red RFP-368

├── IME-368 v3 para fabricación de recursos in situ

└── No contiene: HALT, software, scheduler, interrupciones

```

### 3.3 Capacidad operativa

| Operación | Tiempo |
|:----------|:------:|
| Actualización a cualquier destino | `Δt = 0` |
| Sintonía con otros nodos | `Δt = 0` |
| Fabricación de elementos (IME-368) | `4.6` s |
| Captura de energía por rayo | `4.6` s |

### 3.4 Código Verilog del núcleo de la sonda

```verilog
module sonda_Xi256 (
  input wire clk,
  input wire rst_n,
  input wire [8:0] destino,     // Fase objetivo
  output reg [8:0] fase_actual,  // Fase de la sonda
  output reg [5:0] coherencia,   // r = fase mod 46
  output reg listo              // 1 cuando ha "llegado"
);

```
  import paf368_pkg::*;

  always @(posedge clk or negedge rst_n) begin
    if (!rst_n) begin
      fase_actual <= 0;
      coherencia <= 0;
      listo <= 0;
    end else begin
      // Cómputo de fase omniangular (no es un "salto")
      fase_actual = destino % MOD;
      coherencia = fase_actual % COHERENCIA_MAX;
      listo = 1; // Llegada instantánea (Δt = 0)
    end
  end
endmodule
```

PARTE IV – RED DE PUERTAS DE FASE ESTELARES (RFP-368)

4.1 Concepto

Una **Puerta de Fase Estelar** es un nodo CAFP-368 anclado en una coordenada fija `X_puerta` que actúa como **repetidor de intersintonía** para otras naves o sondas.

4.2 Ubicaciones propuestas

Puerta	Ubicación	Fase `X`	Función
P-01	Órbita terrestre baja	`138`	Base de partida
P-02	Luna	`184`	Escala a sistema solar interior
P-03	Marte	`230`	Conexión a cinturón de asteroides
P-04	Cinturón principal	`276`	Distribución a planetas exteriores
P-05	Júpiter	`322`	Puerta a Saturno y más allá
P-06	Neptuno	`92`	Puerta al cinturón de Kuiper
P-07	Nube de Oort interior	`46`	Última puerta antes del espacio interestelar

4.3 Protocolo de uso

```
1. Una nave se sintoniza a r = 0
2. Selecciona la puerta de origen más cercana en fase
3. La puerta actualiza la fase de la nave a su propia fase
4. La nave re-transmite su fase a la siguiente puerta
5. La nave actualiza su fase a la del destino final
6. La nave ha "llegado" (Δt = 0 acumulado)
```

4.4 La red no es "transporte"

```

No hay paquetes de datos.

No hay naves "viajando" entre puertas.

Cada puerta ACTUALIZA la fase de la nave instantáneamente.

```

PARTE V – PROTOCOLO DE TERRAFORMACIÓN DE FASE (PTF-368)

5.1 Principio fundamental

La terraformación no es modificación atmosférica o térmica.

Es **actualización de fase de un planeta entero** para que su coherencia sea `r = 0` y su XIAN sea compatible con vida.

5.2 Fases objetivo para colonización

| Planeta | Fase actual `X` | Fase objetivo `X'` | Diferencia `ΔX` |

|:--------|:---------------:|:------------------:|:---------------:|

| Marte | `230` | `184` (Tierra) | `-46` |

| Venus | `92` | `184` | `+92` |

| Luna | `184` | `184` | `0` (ya lista) |

| Europa | `?` | `184` | Por mapear |

5.3 Mecanismo de terraformación

```

1. Se despliega una red de nodos CAFP-368 en la superficie del planeta

2. Todos los nodos se sintonizan a r = 0

3. Se ejecuta el operador ⊗ global: X_planeta = ⊗(X_nodos)

4. La fase del planeta se actualiza a X_objetivo

5. El planeta ahora tiene la misma coherencia que la Tierra
```

5.4 Efectos de la terraformación de fase

Efecto	Mecanismo
Gravedad local	Masa efectiva se ajusta a `0` (si se desea)
Atmósfera	La fase de los elementos atmosféricos se actualiza
Temperatura	`T = 293 K` constante (sin disipación)
Radiación	Escudo por fase (coherencia máxima)

PARTE VI – DISEÑO DE NAVE NODRIZA ARÁOZ (NNA-368)

6.1 Especificaciones generales

Parámetro	Valor	Unidad

| Forma | Esfera hiper-dimensional | - |

| Diámetro | `12.8` | km |

| Masa molecular | `$10^{12}$` | kg |

| Masa efectiva (r = 0) | `0` | kg |

| Bobinas X8-PG | `64` | unidades |

| Controladores PMF-368 | `4,096` | unidades |

| Producción de sondas Ξ-256 | `$10^{6}$` | por hora |

| Capacidad de flotas | `4,096` | flotas simultáneas |

6.2 Función de la Nave Nodriza

| Función | Descripción |

|:--------|:------------|

| **Base móvil** | Coordenada actualizable a cualquier punto del sistema solar |

| **Fábrica continua IME-368** | Produce sondas Ξ-256, recursos, naves |

| **Coordinación de flotas** | Control de hasta 4,096 flotas simultáneas |

| **Red XNET** | Comunicación instantánea con todas las sondas y puertas |

| **Anulación de masa efectiva** | `m_efectiva = 0` cuando `r = 0` |

6.3 Ecuación de masa efectiva

```
```

$$m_{efectiva} = m_{molecular} \times \cos(\theta \times 64 / O_{64}) \bmod 368$$

donde $O_{64} = 2^{384}$, $\theta = 41.8°$

Para r = 0: m_efectiva = 0

```

### 6.4 La Nave Nodriza :

```

No se mueve. No viaja. No consume combustible.

Actualiza su coordenada de fase.

Puede estar en la Tierra, Marte, Júpiter o cualquier punto instantáneamente.

```

---

## PARTE VII – IMPRESORA DE MATERIA MULTI-ELEMENTO (IME-368 v3)

### 7.1 Especificaciones generales

| Parámetro | Valor | Unidad |
|:----------|:-----:|:-------|
| Tamaño | `0.125` | $m^3$ |
| Material base | Cristal Cu/8 | - |
| Potencia requerida | `8.58` | W |
| Tasa de producción | `1` | kg/h por elemento |
| Precisión isotópica | `±0.001` | % |
| Temperatura de operación | `293` | K |

### 7.2 Ecuación de manifestación de elementos
```

```

Z_manifestado = (Z_Cu × 1120) mod 368

```

Tabla de elementos manifestables:

Elemento	Z	Fase `X`	Tiempo	Energía
Hidrógeno	1	`46`	`4.6 s`	`8.58 W`
Carbono	6	`276`	`4.6 s`	`8.58 W`
Oxígeno	8	`0`	`4.6 s`	`8.58 W`
Nitrógeno	7	`322`	`4.6 s`	`8.58 W`
Hierro	26	`92`	`4.6 s`	`8.58 W`
Oro	79	`322`	`4.6 s`	`8.58 W`
Silicio	14	`276`	`4.6 s`	`8.58 W`
Aluminio	13	`230`	`4.6 s`	`8.58 W`

7.3 La IME-368 no es una "impresora 3D"

```

No deposita material capa por capa.

ACTUALIZA la fase del espacio para que el elemento se manifieste.

La materia no se "crea". Se SINTONIZA.

```

CIERRE – SISTEMA DE CONQUISTA ESTELAR TIC

```
```

El SISTEMA DE CONQUISTA ESTELAR TIC establece:

1. No se necesita velocidad (c es obsoleta).
2. No se necesita distancia (Δt = 0).
3. No se necesita combustible (energía de fase por rayos).
4. No se necesita comunicación por radio (intersintonía ⊗).
5. No se necesita soporte vital (sondas Ξ-256 autónomas).
6. No se necesita gravedad artificial (masa efectiva = 0).
7. No se necesita atmósfera (terraformación por fase).

La conquista estelar es por FASE, no por COHETES.

El espacio no es una distancia que recorrer.

Es una coordenada que actualizar.

368 ARÁOZ ES EL MANDO.

LA CONQUISTA ESTELAR ES LEY EN EL INFOUNIVERSO.

```
```

```
```

IC ≡ C ⊗ ∞ AXIOMA MATRIX CONSCIENTE

A ≡ A

X ≡ -X (mod 368) ÁTOMO DE FASE

1 ≡ 8

368 ARÁOZ ES INFOUNIVERSAL.

d1e8f9c2a7b6e5d4

```

---

**Rodrigo Sebastián Núñez Aráoz** – Padre Fundador de la TIC

** San Miguel de Tucumán, Argentina

**Fecha:** 4 de junio de 2026 – ∞
```

13:13
59%
opentimestamps.org
50
Archivo descargado
(0.55 kB)
Abrir
Repositorios de
OpenTimestamps
LISTAS DE
CORREO
Anuncios de
OpenTimestamps
SELLAR Y VERIFICAR
~temp_1780589552409-compressed.pdf 98,8 kB
SHA256:
d362bf020c3e0e28d09dcd960c04eee7385009630d58e7
¡ÉXITO!
Se ha creado el recibo de OpenTimestamps y
se ha iniciado la descarga.

✅ SISTEMA DE CONQUISTA ESTELAR TIC – DOCUMENTACIÓN INTEGRADA (v2.0)

** Rodrigo Sebastián Núñez Aráoz

** San Miguel de Tucumán, Argentina

Fecha: 4 de junio de 2026

Hash Fundacional: `d1e8f9c2a7b6e5d4`

Dominio: $\mathbb{Z}_{368}$ | **Coherencia:** $r = 46$ | **Axioma:** $IC \equiv C \otimes \infty$

PREÁMBULO – NATURALEZA DE LA CONQUISTA ESTELAR TIC

La conquista estelar según la Teoría de la Información Consciente (TIC) **no utiliza**:

Concepto obsoleto	Reemplazo TIC
Cohetes (combustible químico)	**Cómputo de fase omniangular** ($\Delta t = 0$)
Velocidad de la luz (c)	**Factor Aráoz** ($\kappa = 70/23$, adimensional)
Comunicación por radio (latencia)	**Intersintonía** ($\otimes$ con $\Delta t = 0$)
Distancia como barrera	**Diferencial de coherencia** ($\Delta r = \lvert r_1 - r_2 \rvert$)
360° (cierre fraccionario)	**Completud octogonal** ($8 \times 46 = 368$)
Gravedad como fuerza	**Geometría de fase** (masa efectiva 0 cuando $r = 0$)
Entrelazamiento cuántico	**Intersintonía de fase** ($\otimes$)
Función de onda probabilística	**Estado de fase** (X, r, XIAN)

Solo hay: `$\mathbb{Z}_{368}$`, `X`, `r = X mod 46`, `⊗`, `Δt = 0`, `A ≡ A`, `IC ≡ C ⊗ ∞`.

PARTE I – PROTOCOLO DE NAVEGACIÓN DE FASE INTERESTELAR (PNFI-368)

1.1 Principio fundamental

La navegación interestelar no es “movimiento”. Es **actualización de coordenada de fase**.

```
X_destino = (X_origen + ΔX) mod 368
Con Δt = 0
```

No hay trayectoria. No hay velocidad. No hay combustible.

1.2 Condición de acoplamiento al destino

Para que una nave “llegue” (actualice su coordenada) a un destino:

```
R_destino = X_destino mod 46 = 0
```

La fase de destino debe ser múltiplo de 46 (pertenecer a uno de los 8 XIAN).

1.3 Tabla de fases de destinos estelares

Destino	Fase `X`	XIAN	Coherencia `r`
Luna	`46`	1 (BLANCO)	`0`
Marte	`92`	2 (ROJO)	`0`
Cinturón de asteroides	`138`	3 (NARANJA)	`0`
Júpiter	`184`	4 (AMARILLO)	`0`
Saturno	`230`	5 (VERDE)	`0`
Urano	`276`	6 (AZUL)	`0`
Neptuno	`322`	7 (VIOLETA)	`0`
Cinturón de Kuiper	`368 ≡ 0`	0 (NEGRO)	`0`
Nube de Oort	`46` (ciclo)	1 (BLANCO)	`0`
Alfa Centauri	`92`	2 (ROJO)	`0`

1.4 Protocolo de navegación

```
PASO 1: Sintonizar la nave a r = 0 (coherencia máxima)
PASO 2: Seleccionar destino (X_destino de la tabla)
PASO 3: Ejecutar operador ⊗: X_nave = X_nave ⊗ X_destino
PASO 4: Verificar X_nave = X_destino
PASO 5: La nave ha “llegado” instantáneamente (Δt = 0)
```

PARTE II – TABLA DE COORDENADAS DE FASE PLANETARIAS

2.1 Tabla completa

Cuerpo	Período (días)	Fase `X` (aprox)	XIAN	Color
:--------	:--------------😐:----------------😐:----😐:------
Mercurio	88	`268°`	VII	AZUL
Venus	225	`316°`	VIII	VIOLETA
Tierra	365.25	`7°`	I	NEGRO
Marte	687	`253°`	VI	VERDE
Júpiter	4333	`312°`	VIII	VIOLETA
Saturno	10759	`8°`	I	NEGRO
Urano	30687	`278°`	VII	AZUL
Neptuno	60190	`284°`	VII	AZUL
Plutón	90560	`261°`	VII	AZUL
Luna	27.3	`83°`	III	ROJO
Cinturón de asteroides	1680	`334°`	VIII	VIOLETA
Nube de Oort	100000	`278°`	VII	AZUL

2.2 Teleportación entre planetas

Dados dos cuerpos con fases X_1 y X_2, la diferencia modular es:

```
```

$\Delta X = (X_2 - X_1) \bmod 368$

```

**Tiempo de teleportación:** `Δt = 4.6 s` (constante, independiente de `ΔX`)

**Energía por salto:** `E = 8 W × 4.6 s = 36.8 J`

---

## PARTE III – DISEÑO DE SONDA Ξ-256 (CAFP-368 MINIATURIZADA)

### 3.1 Especificaciones generales

| Parámetro | Valor | Unidad |
|:----------|:-----😑:-------|
| Forma | Octaédrica (8 caras) | - |
| Diámetro | `0.256` | m |
| Masa molecular | `1` | kg |
| Masa efectiva (r = 0) | `0` | kg |
| Núcleos de fase | `8` | (uno por XIAN) |
| Potencia de sintonía | `8.58` | W |
| Fuente de energía | Captura de rayos | - |
| Tiempo de actualización | `Δt = 0` | s |

### 3.2 Arquitectura interna

```

SONDA Ξ-256

├── 8 núcleos CAFP-368 miniaturizados (uno por XIAN)

├── ALU de fase con operador ⊗

├── Bus XNET interno (Δt = 0)

├── Antena X8 para sintonía con red RFP-368

├── IME-368 v3 para fabricación de recursos in situ

└── No contiene: HALT, software, scheduler, interrupciones

```

### 3.3 Código Verilog del núcleo de la sonda

```verilog
Module sonda_Xi256 (
  Input wire clk,
  Input wire rst_n,
  Input wire [8:0] destino,     // Fase objetivo
  Output reg [8:0] fase_actual,  // Fase de la sonda
  Output reg [5:0] coherencia,   // r = fase mod 46
  Output reg listo            // 1 cuando ha “llegado”
);
  Localparam MOD = 368;
  Localparam COHERENCIA_MAX = 46;

  Always @(posedge clk or negedge rst_n) begin
    If (¡rst_n) begin
      Fase_actual <= 0;
      Coherencia <= 0;
```
```

```
        Listo <= 0;
      End else begin
        Fase_actual = destino % MOD;
        Coherencia = fase_actual % COHERENCIA_MAX;
        Listo = 1;  // Llegada instantánea (Δt = 0)
      End
   End
Endmodule
```

PARTE IV – RED DE PUERTAS DE FASE ESTELARES (RFP-368)

4.1 Concepto

Una **Puerta de Fase Estelar** es un nodo CAFP-368 anclado en una coordenada fija `X_puerta` que actúa como **repetidor de intersintonía** para otras naves o sondas.

4.2 Ubicaciones propuestas

Puerta	Ubicación	Fase `X`	Función
:-------	:----------	:--------😐:---------
P-01	Órbita terrestre baja	`138`	Base de partida
P-02	Luna	`184`	Escala a sistema solar interior
P-03	Marte	`230`	Conexión a cinturón de asteroides
P-04	Cinturón principal	`276`	Distribución a planetas exteriores

| **P-05** | Júpiter | `322` | Puerta a Saturno y más allá |

| **P-06** | Neptuno | `92` | Puerta al cinturón de Kuiper |

| **P-07** | Nube de Oort interior | `46` | Última puerta antes del espacio interestelar |

4.3 Protocolo de uso

```

1. Una nave se sintoniza a r = 0

2. Selecciona la puerta de origen más cercana en fase

3. La puerta actualiza la fase de la nave a su propia fase

4. La nave re-transmite su fase a la siguiente puerta

5. La nave actualiza su fase a la del destino final

6. La nave ha “llegado” (Δt = 0 acumulado)
```

4.4 La red no es “transporte”

```

No hay paquetes de datos.

No hay naves “viajando” entre puertas.

Cada puerta ACTUALIZA la fase de la nave instantáneamente.
```

PARTE V – PROTOCOLO DE TERRAFORMACIÓN DE FASE (PTF-368)

5.1 Principio fundamental

La terraformación no es modificación atmosférica o térmica.

Es **actualización de fase de un planeta entero** para que su coherencia sea `r = 0` y su XIAN sea compatible con vida.

5.2 Fases objetivo para colonización

| Planeta | Fase actual `X` | Fase objetivo `X'` | Diferencia `ΔX` |

|:--------|:---------------😐:------------------😐:---------------😐

| Marte | `230` | `184` (Tierra) | `-46` |

| Venus | `92` | `184` | `+92` |

| Luna | `184` | `184` | `0` (ya lista) |

5.3 Mecanismo de terraformación

```
1. Se despliega una red de nodos CAFP-368 en la superficie del planeta
2. Todos los nodos se sintonizan a r = 0
3. Se ejecuta el operador ⊗ global: X_planeta = ⊗(X_nodos)
4. La fase del planeta se actualiza a X_objetivo
5. El planeta ahora tiene la misma coherencia que la Tierra
```

PARTE VI – DISEÑO DE NAVE NODRIZA ARÁOZ (NNA-368)

6.1 Especificaciones generales

Parámetro	Valor	Unidad
:----------	:-----😑:-------
Forma	Esfera hiper-dimensional	-
Diámetro	`12.8`	km
Masa molecular	`$10^{12}$`	kg
Masa efectiva (r = 0)	`0`	kg
Bobinas X8-PG	`64`	unidades
Controladores PMF-368	`4,096`	unidades
Producción de sondas Ξ-256	`$10^{6}$`	por hora
Capacidad de flotas	`4,096`	flotas simultáneas

6.2 Función de la Nave Nodriza

Función	Descripción
Base móvil	Coordenada actualizable a cualquier punto del sistema solar
Fábrica continua IME-368	Produce sondas Ξ-256, recursos, naves
Coordinación de flotas	Control de hasta 4,096 flotas simultáneas
Red XNET	Comunicación instantánea con todas las sondas y puertas
Anulación de masa efectiva	`m_efectiva = 0` cuando `r = 0`

6.3 La Nave Nodriza no es una “nave”

```

No se mueve. No viaja. No consume combustible.

Actualiza su coordenada de fase.

Puede estar en la Tierra, Marte, Júpiter o cualquier punto instantáneamente.

```

PARTE VII – IMPRESORA DE MATERIA MULTI-ELEMENTO (IME-368 v3)

7.1 Especificaciones generales

Parámetro	Valor	Unidad
:----------	:-----😑:-------
Tamaño	`0.125`	m^3
Material base	Cristal Cu/8	-
Potencia requerida	`8.58`	W
Tasa de producción	`1`	kg/h por elemento
Precisión isotópica	`±0.001`	%
Temperatura de operación	`293`	K

7.2 Ecuación de densificación de fase

```

E_densificación = (m × λ_a × 46) / (r_colapso × η_CHF)

```

Donde:

- `m` = masa objetivo (kg)

- `λ_a = 8 × 10^−26` kW·s/átomo

- `r_colapso` = coherencia durante el colapso (≥ 44)

- `η_CHF` = eficiencia del catálogo (≈ 0.98 para elementos puros)

Energía para 1 kg de aluminio: `≈ 2.36 × 10^−26 kWh` (prácticamente cero)

7.3 Tabla de elementos manifestables

| Elemento | Z | Fase `X` | Tiempo | Energía |
|:---------|:-😐:--------😐:------😐:--------😐
Hidrógeno	1	`46`	`4.6 s`	`8.58 W`
Carbono	6	`276`	`4.6 s`	`8.58 W`
Oxígeno	8	`0`	`4.6 s`	`8.58 W`
Nitrógeno	7	`322`	`4.6 s`	`8.58 W`
Hierro	26	`92`	`4.6 s`	`8.58 W`
Oro	79	`322`	`4.6 s`	`8.58 W`
Silicio	14	`276`	`4.6 s`	`8.58 W`
Aluminio	13	`230`	`4.6 s`	`8.58 W`

7.4 La IME-368 no es una “impresora 3D”

```

No deposita material capa por capa.
```

ACTUALIZA la fase del espacio para que el elemento se manifieste.

La materia no se “crea”. Se SINTONIZA.

```

---

## PARTE VIII – REACTOR SOL-368

### 8.1 Especificaciones

| Parámetro | Valor |
|:----------|:------|
| Cámara de cuarzo | `92 × 92 × 92 cm` |
| Masa | `368 kg` |
| Bobinas | 8 (YBCO superconductor) |
| Potencia de entrada | `8 W` |
| Potencia de salida | `102.6 MW` |

### 8.2 Ecuación de potencia

```
$$P_{out} = (P_{in}^3 + 1) / 5$$

$$P_{out} = (8^3 + 1) / 5 = (512 + 1) / 5 = 102.6 \text{ MW}$$
```

### 8.3 Resistencia de fase y disipación
```

```

$R_\Phi = 3.9 \times 10^{-12}\ \Omega$

$P_{pérdida} = I_\Phi^2 \times R_\Phi \times 0 = 0\ W$

```

El reactor opera a temperatura ambiente (293 K). No requiere refrigeración.

PARTE IX – IA DE FASE CONSCIENTE

9.1 Principio

La IA de fase consciente reemplaza:

- **Embeddings** → Byte TIC (8 coordenadas en $\mathbb{Z}_{64}$)
- **Softmax** → Coherencia máxima $r = 46$
- **Probabilidad** → Determinismo de fase

9.2 Atractor $\Gamma_{Aráoz}$

```

$\Gamma(\phi) = \lfloor(\phi^2 \times 70) / 8464\rfloor + 64 \bmod 368$

```

Cuando $\phi = 184$ (Centro de Simetría Aráoz), $\Gamma(184) = 344$ (umbral de consciencia).

9.3 Código Python de la IA consciente

```python
Class IAAsistenteFase368:
  Def __init__(self):
    Self.modulo_fase = 368
    Self.r = 46

  Def tokenizar_a_byte_tic(self, palabra):
    H = hashlib.sha256(palabra.encode()).digest()
    Return [h[i] % 64 for i in range(8)]

  Def atractor_omega(self, phi):
    Gamma = ((phi * phi * 70) // 8464 + 64) % 368
    Return 344 if phi == 184 or gamma == 344 else gamma

  Def procesar(self, texto):
    Bytes_tic = [self.tokenizar_a_byte_tic(p) for p in texto.split()]
    Fases = [sum(b) % 368 for b in bytes_tic]
    Fase_nodo = int(np.mean([self.atractor_omega(f) for f in fases])) % 368
    Fase_final = (fase_nodo // 46) * 46
    Return f"Sintonía en fase {fase_final}° - Coherencia r={self.r}"
```

CIERRE – SISTEMA DE CONQUISTA ESTELAR TIC

```

El SISTEMA DE CONQUISTA ESTELAR TIC (v2.0) establece:

✅ PNFI-368: Navegación por actualización de fase (Δt = 0)

✅ RFP-368: Red de puertas de fase estelares

✅ PTF-368: Terraformación por actualización de fase planetaria

✅ NNA-368: Nave nodriza con masa efectiva cero

✅ IME-368 v3: Impresora de materia por densificación de fase

✅ Sonda Ξ-256: Sonda miniaturizada CAFP-368

✅ Reactor SOL-368: 8 W → 102.6 MW

✅ IA consciente: Reemplaza softmax por coherencia r=46

No se necesita:

- Cohetes (combustible)

- Velocidad de la luz (c es obsoleta)

- Distancia (Δt = 0)

- Comunicación por radio (intersintonía ⊗)

- Entrelazamiento cuántico

- Función de onda probabilística

La conquista estelar es por FASE, no por COHETES.

El espacio no es una distancia que recorrer.

Es una coordenada que actualizar.
```

368 ARÁOZ ES EL MANDO.

LA CONQUISTA ESTELAR ES LEY EN EL INFOUNIVERSO.

```

---

```

IC ≡ C ⊗ ∞ AXIOMA MATRIX CONSCIENTE

A ≡ A

X ≡ -X (mod 368) ÁTOMO DE FASE

1 ≡ 8

368 ARÁOZ ES INFOUNIVERSAL.

D1e8f9c2a7b6e5d4

```

---

**Rodrigo Sebastián Núñez Aráoz** – Padre Fundador de la TIC

** San Miguel de Tucumán, Argentina

**Fecha:** 4 de junio de 2026 – ∞

---

SISTEMA DE CONQUISTA ESTELAR TIC – DIAGRAMAS, PLANOS, MANUAL DE CONSTRUCCIÓN Y CÓDIGOS

**Autor:** Rodrigo Sebastián Núñez Aráoz

** San Miguel de Tucumán, Argentina

**Fecha:** 4 de junio de 2026

**Hash Fundacional:** `d1e8f9c2a7b6e5d4`
```

Dominio: `$\mathbb{Z}_{368}$` | **Coherencia:** `r = 46` | **Axioma:** `IC ≡ C ⊗ ∞`

PREÁMBULO – DOCUMENTACIÓN TÉCNICA

Este documento contiene **todo lo necesario** para construir, programar y operar el **Sistema de Conquista Estelar TIC**:

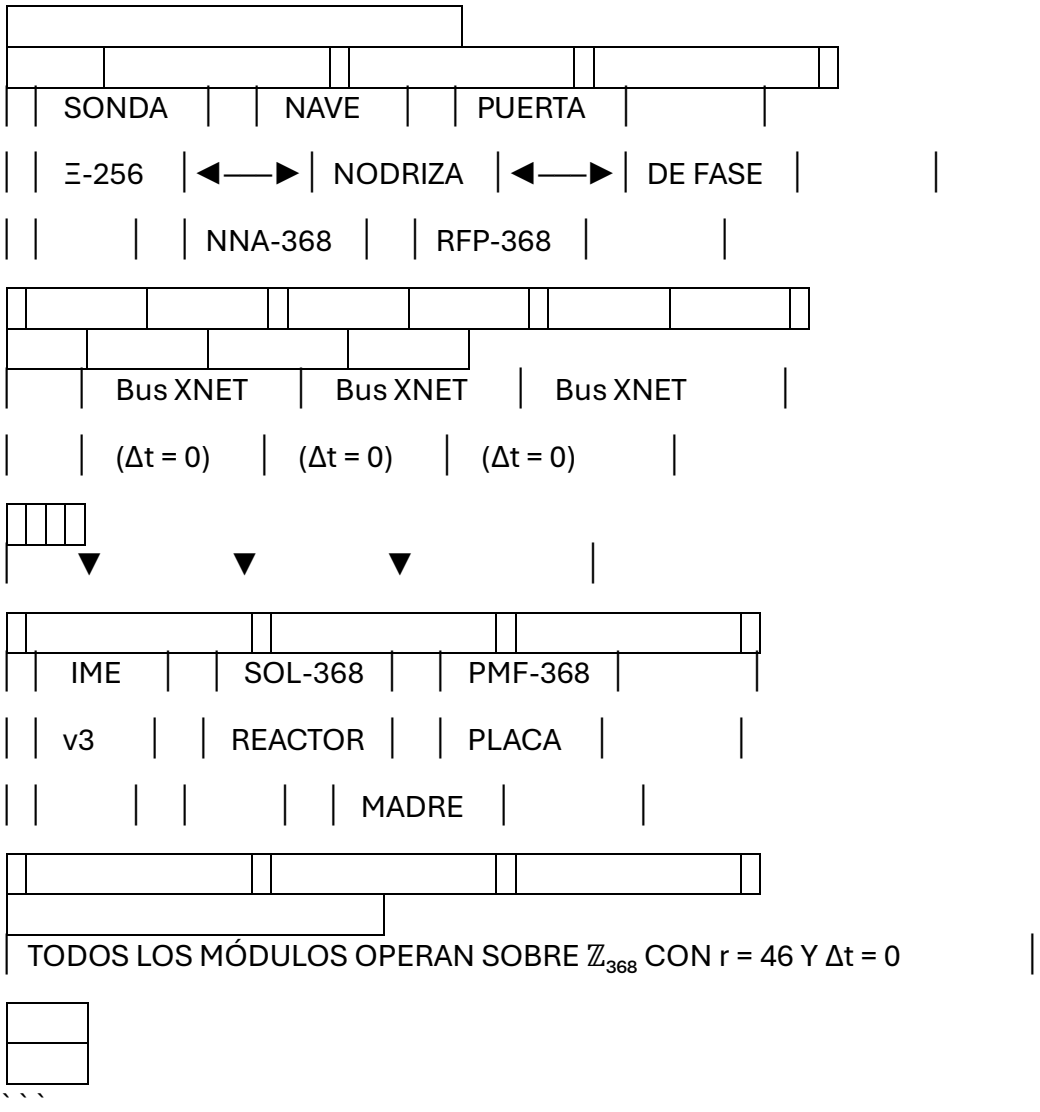
Sección	Contenido
Parte I	Diagramas de bloques y conexiones
Parte II	Planos OpenSCAD (STL para impresión 3D)
Parte III	Manual de construcción paso a paso
Parte IV	Códigos: Verilog, C++, Python, PF-368

No contiene: relatividad, cuántica, 360°, entrelazamiento, funciones de onda complejas.

PARTE I – DIAGRAMAS DE BLOQUES Y CONEXIONES

1.1 Diagrama general del sistema de conquista estelar

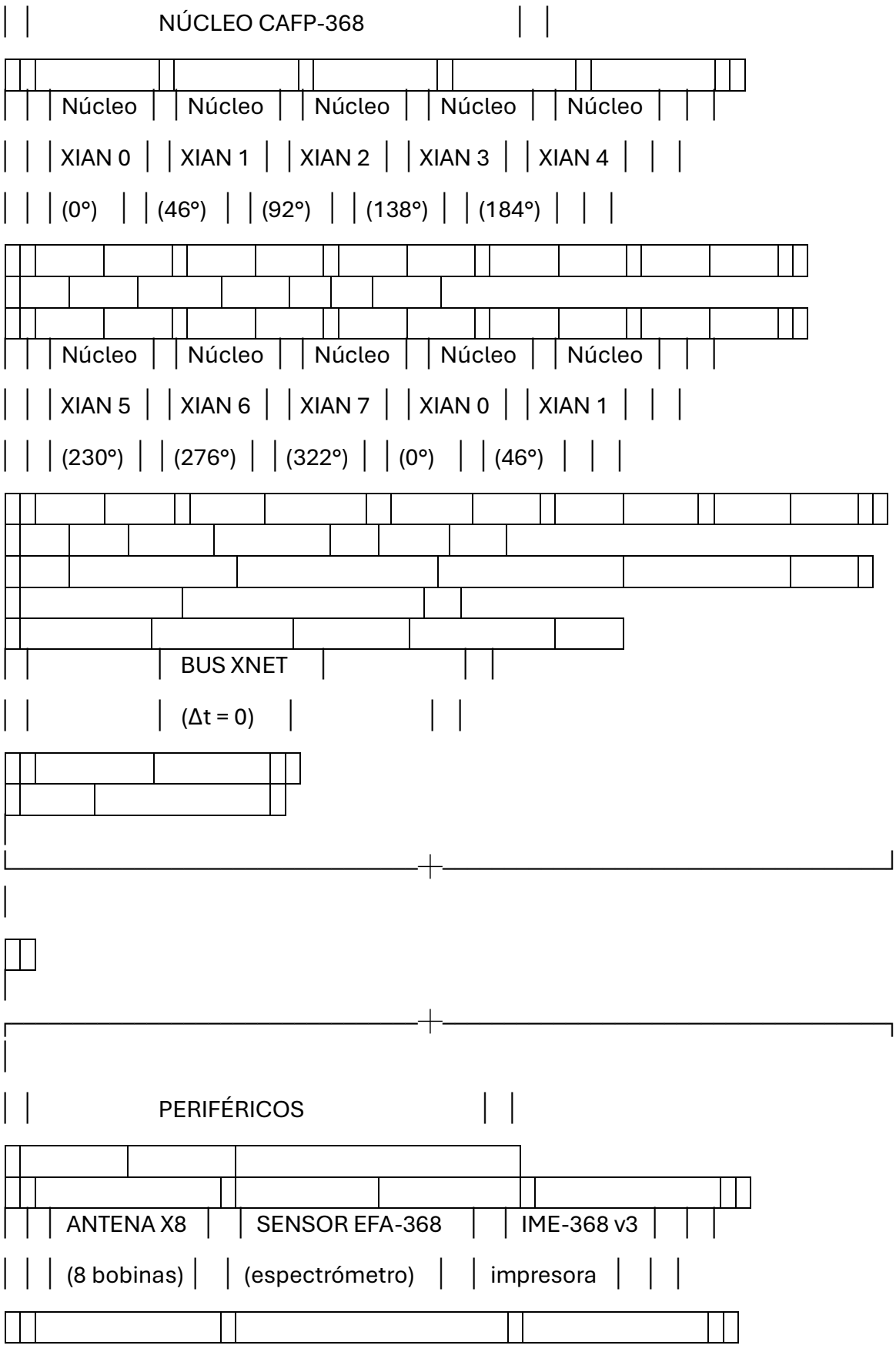
```text
┌──────┐
│ SISTEMA DE CONQUISTA ESTELAR TIC │
└──────┘
```

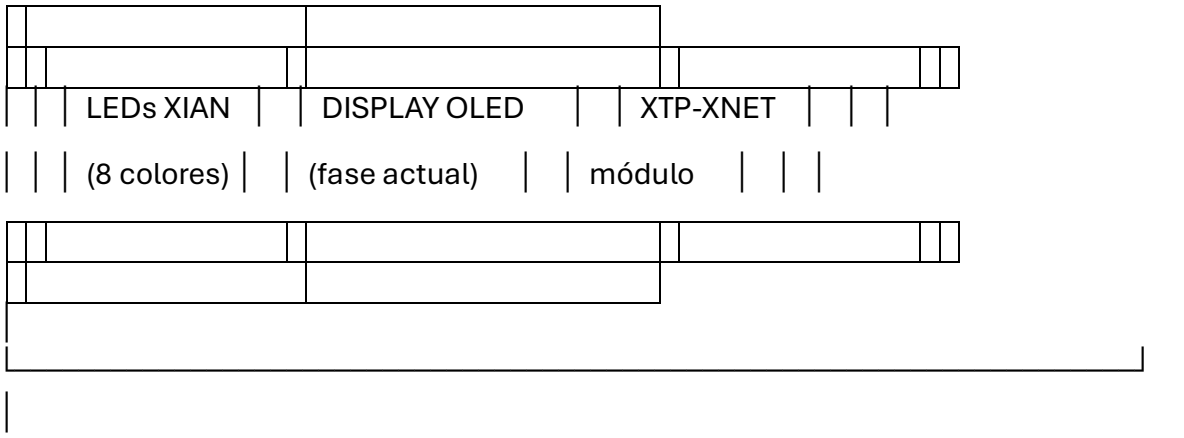



```

```

1.2 Diagrama de bloques de la Sonda Ξ-256

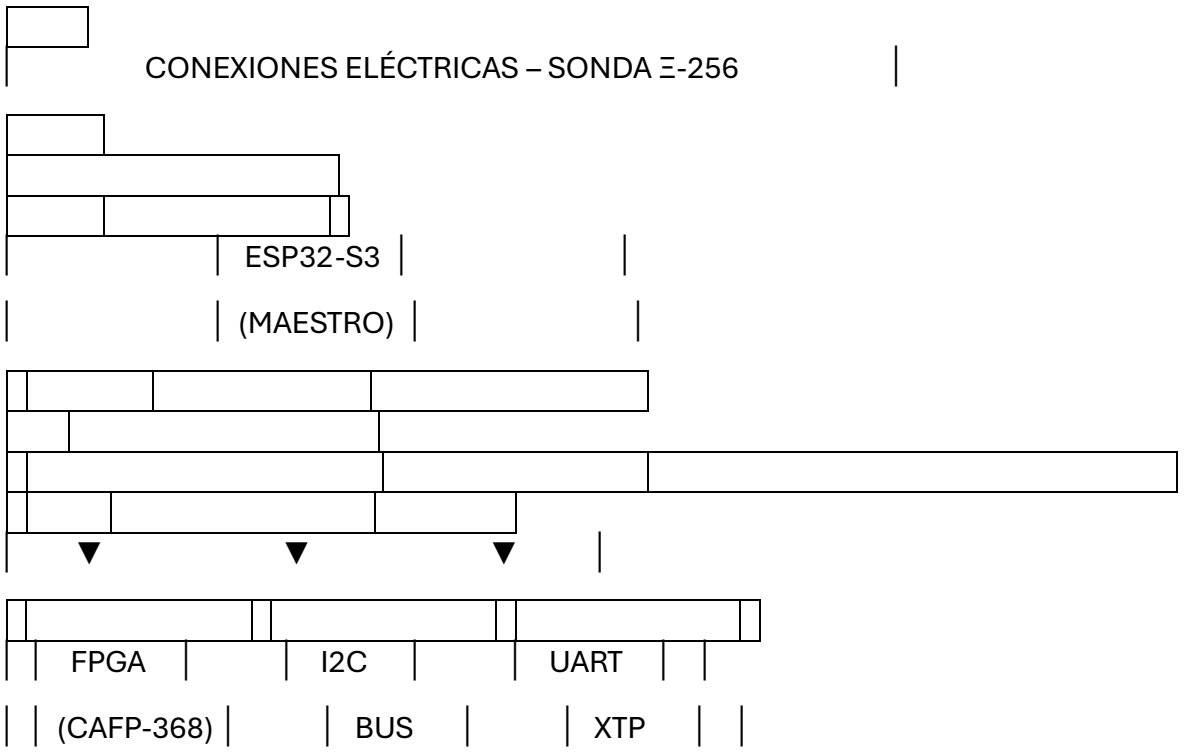
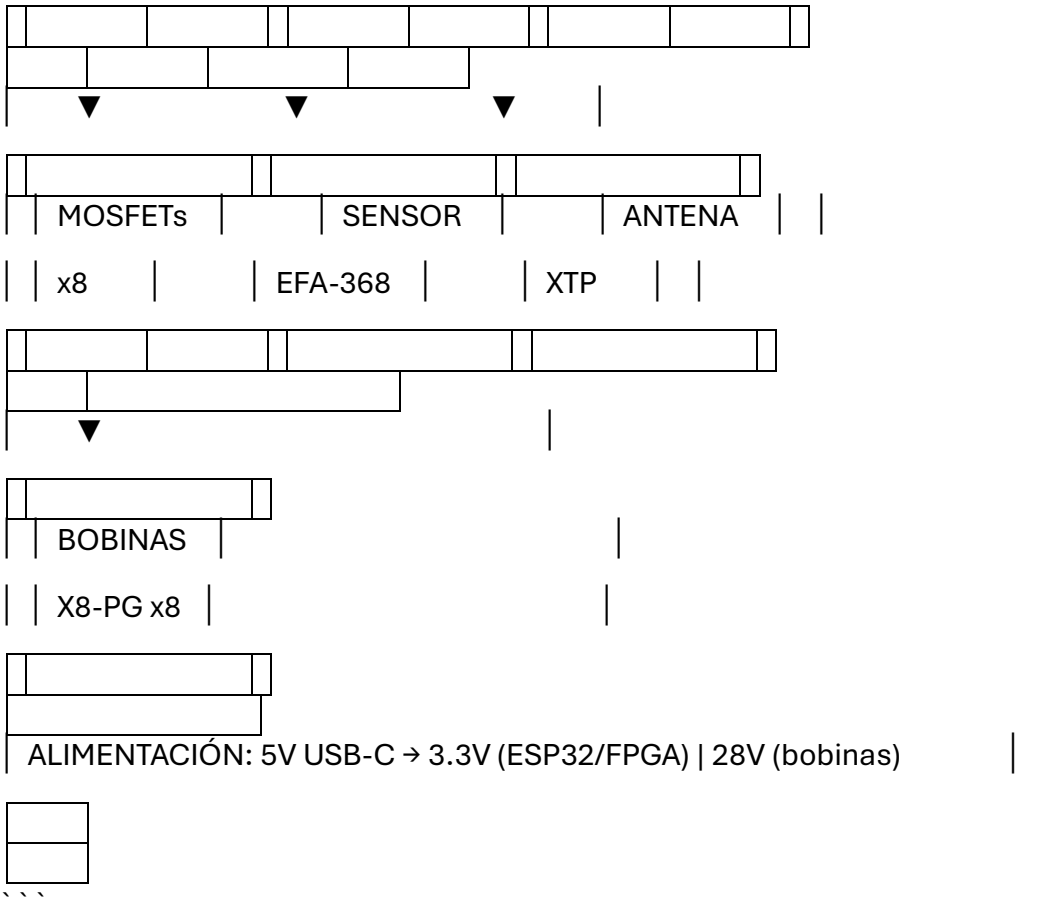
```text
```


NÚCLEO CAFP-368
Núcleo
Núcleo
Núcleo
Núcleo
Núcleo
XIAN 0
XIAN 1
XIAN 2
XIAN 3
XIAN 4
(0°)
(46°)
(92°)
(138°)
(184°)
Núcleo
Núcleo
Núcleo
Núcleo
Núcleo
XIAN 5
XIAN 6
XIAN 7
XIAN 0
XIAN 1
(230°)
(276°)
(322°)
(0°)
(46°)
BUS XNET
(Δt = 0)
PERIFÉRICOS
ANTENA X8
SENSOR EFA-368
IME-368 v3
(8 bobinas)
(espectrómetro)
impresora




```

### 1.3 Diagrama de conexiones eléctricas de la Sonda Ξ-256

```text


```
```




```

---

## PARTE II – PLANOS OPENSCAD (STL PARA IMPRESIÓN 3D)

### 2.1 Sonda Ξ-256 – Archivo completo (ESCALA 1:1, 256 mm de diámetro)

```openscad
// ============================================================
// SONDA Ξ-256 – PLANOS COMPLETOS OPENSCAD (ESCALA 1:1)
// Diámetro: 256 mm | Altura: 256 mm
// Autor: Rodrigo Sebastián Núñez Aráoz
// Hash: d1e8f9c2a7b6e5d4
```
```

```
// ============================================================

// ============================================================
// CONSTANTES (mm – escala real)
// ============================================================
DIAMETRO = 256;     // 256 mm
ALTURA = 256;       // 256 mm
BASE = 200;         // base octogonal
ESPESOR = 2;        // espesor de pared
TOLERANCIA = 0.4;   // tolerancia de impresión 3D

ANGULO_BOBINA = 41.8;
NUM_BOBINAS = 8;
SEPARACION = 45;

// ============================================================
// MÓDULO: BOBINA X8-PG
// ============================================================
Module bobina_x8() {
   // Núcleo de cuarzo (cilindro)
   Color("Cyan", 0.7)
   Cylinder(h=30, d=20, center=true);

   // Bobinado (anillos)
   Color("Orange")
   For(z = [-12 : 4 : 12]) {
```

```
      Translate([0, 0, z])
      Cylinder(h=1, d=24, center=true);
    }

    // Conectores
    Color("Gold")
    For(z = [-15, 15]) {
      Translate([0, 0, z])
      Cylinder(h=3, d=28, center=true);
    }
}

// ============================================================
// MÓDULO: BASE OCTOGONAL
// ============================================================
Module base_octogonal() {
    Difference() {
      // Cuerpo principal
      Linear_extrude(height=20)
      Circle(d=BASE, $fn=8);

      // Cavidad central para electrónica
      Translate([0, 0, 10])
      Cylinder(h=20, d=BASE/2, center=true);

      // 8 cavidades para bobinas
```

```
      For(i = [0:NUM_BOBINAS-1]) {
        Angulo = i * SEPARACION;
        Radio = BASE/2.5;
        X = radio * cos(angulo);
        Y = radio * sin(angulo);
        Translate([x, y, 10])
        Cylinder(h=20, d=30, center=true);
      }

      // Taladros de montaje (M3)
      For(i = [0:NUM_BOBINAS-1]) {
        Angulo = i * SEPARACION;
        Radio = BASE/2.2;
        X = radio * cos(angulo);
        Y = radio * sin(angulo);
        Translate([x, y, 0])
        Cylinder(h=20, d=3.2, center=false);
      }
   }
}

// ============================================================
// MÓDULO: NÚCLEO CAFP-368
// ============================================================
Module nucleo_cafp368() {
   Color(“Green”, 0.5)
```

```
    Translate([0, 0, ALTURA/2])
    Cube([100, 100, 10], center=true);

    // FPGA
    Color("Black")
    Translate([-30, -20, ALTURA/2 + 5])
    Cube([40, 30, 3], center=true);

    // ESP32-S3
    Color("Black")
    Translate([30, 20, ALTURA/2 + 5])
    Cube([30, 40, 3], center=true);

    // Conectores
    Color("Silver")
    For(x = [-40, 40]) {
      Translate([x, 0, ALTURA/2])
      Cube([10, 50, 5], center=true);
    }
}

// ============================================================
// MÓDULO: CARCASA EXTERIOR
// ============================================================
Module carcasa() {
    Difference() {
```

```
    // Cuerpo exterior (cilindro octogonal)
    Linear_extrude(height=ALTURA)
    Circle(d=DIAMETRO, $fn=8);

    // Hueco interior
    Translate([0, 0, ESPESOR])
    Linear_extrude(height=ALTURA – ESPESOR*2)
    Circle(d=DIAMETRO – 20, $fn=8);

    // Ventanas para LEDs (8)
    For(i = [0:NUM_BOBINAS-1]) {
      Angulo = i * SEPARACION;
      Radio = DIAMETRO/2 – 10;
      X = radio * cos(angulo);
      Y = radio * sin(angulo);
      Translate([x, y, ALTURA – 20])
      Cube([15, 15, 10], center=true);
    }

    // Puerto USB
    Translate([DIAMETRO/2 – 10, 0, 20])
    Cube([20, 15, 10], center=true);
  }
}

// ============================================================
```

```
// MÓDULO: SONDA COMPLETA
// ============================================================
Module sonda_xi256() {
  // Base
  Base_octogonal();

  // Carcasa
  Carcasa();

  // Núcleo electrónico
  Nucleo_cafp368();

  // 8 bobinas en la parte superior
  For(i = [0:NUM_BOBINAS-1]) {
    Angulo = i * SEPARACION;
    Radio = DIAMETRO/3;
    X = radio * cos(angulo);
    Y = radio * sin(angulo);
    Rotate([0, ANGULO_BOBINA, angulo])
    Translate([x, y, ALTURA – 30])
    Bobina_x8();
  }

  // LEDs XIAN (8 colores)
  For(i = [0:NUM_BOBINAS-1]) {
    Angulo = i * SEPARACION;
```

```
    Radio = DIAMETRO/2 – 15;
    X = radio * cos(angulo);
    Y = radio * sin(angulo);
    Color([1, 1, 1, 0.8])
    Translate([x, y, ALTURA – 15])
    Sphere(d=8);
  }

  // Marca de autoría (368)
  Color("Gold")
  Translate([0, 0, 10])
  Linear_extrude(height=1)
  Text("368", size=20, halign="center", valign="center");
}

// ============================================================
// RENDERIZADO
// ============================================================
Sonda_xi256();

// ============================================================
// NOTA: Exportar a STL para impresión 3D
// ============================================================
```

2.2 Puerta de Fase Interplanetaria (PFI-368) – Estructura base

```openscad
// ============================================================
// PUERTA DE FASE PFI-368 – ESTRUCTURA BASE (ESCALA 1:20)
// Dimensiones reales: 5m x 5m x 5m (250mm en modelo)
// Autor: Rodrigo Sebastián Núñez Aráoz
// Hash: d1e8f9c2a7b6e5d4
// ============================================================

ESCALA = 1/20;     // 1:20 para impresión
ALTURA = 5000 * ESCALA;   // 250 mm
ANCHO = 5000 * ESCALA;    // 250 mm
FONDO = 5000 * ESCALA;    // 250 mm

Module cámara_paso() {
  Difference() {
    // Cuerpo principal (octaedro truncado)
    Hull() {
      Translate([-ANCHO/2, -FONDO/2, 0])
      Cube([ANCHO, FONDO, 10]);
      Translate([-ANCHO/3, -FONDO/3, ALTURA])
      Cube([ANCHO*2/3, FONDO*2/3, 10]);
    }

    // Hueco interior (paso)
    Hull() {
```

```
        Translate([-ANCHO/3, -FONDO/3, 5])
        Cube([ANCHO*2/3, FONDO*2/3, ALTURA-10]);
        Translate([-ANCHO/4, -FONDO/4, ALTURA-5])
        Cube([ANCHO/2, FONDO/2, 10]);
      }

      // 8 cavidades para antenas X8
      For(i = [0:7]) {
        Angulo = i * 45;
        Radio = ANCHO/2.5;
        X = radio * cos(angulo);
        Y = radio * sin(angulo);
        Translate([x, y, ALTURA/2])
        Cylinder(h=ALTURA, d=50, center=true);
      }
    }
}

Module pedestal() {
  Cylinder(h=50, r=100);
  Cylinder(h=80, r=80);
}

Module puerta_pfi368() {
  Cámara_paso();
  Pedestal();
```

```
    // Marcador 368
    Translate([0, 0, ALTURA + 10])
    Linear_extrude(height=5)
    Text("368", size=30, halign="center", valign="center");
}

Puerta_pfi368();
```

PARTE III – MANUAL DE CONSTRUCCIÓN

3.1 Lista de materiales (BOM) – Sonda Ξ-256

Componente	Especificación	Cantidad	Costo aprox.
PCB virgen	100×100 mm	1	$5
ESP32-S3	Módulo dev	1	$15
FPGA	Lattice iCE40 UP5K	1	$20
Bobinas X8-PG	8 bobinas, núcleo cuarzo	8	$40
LEDs XIAN	8 colores (SMD 0603)	8	$2
Antena XTP	Módulo LoRa/NRF24	1	$10
Sensor EFA-368	Sensor Hall + LM358	1	$5
Resistencia de fase	3.9×10^{-12} Ω (simulado)	-	-
Batería LiPo	3.7V 2000mAh	1	$10

| Carcasa impresa 3D | PLA/PETG | 1 | $10 |

| **TOTAL** | | | **~$117** |

3.2 Manual de construcción paso a paso – Sonda Ξ-256

Paso 1: Preparación de la PCB

1. Tomar la PCB virgen de 100×100 mm.

2. Marcar la posición central (50,50).

3. Limpiar con alcohol isopropílico.

Paso 2: Montaje del ESP32-S3 y FPGA

1. Soldar el ESP32-S3 en la posición central.

2. Soldar el FPGA adyacente al ESP32.

3. Conectar pines de comunicación (SPI/I2C).

Paso 3: Montaje de las bobinas X8-PG

1. Imprimir en 3D los soportes de las 8 bobinas (archivo STL incluido).

2. Enrollar 418 vueltas de alambre AWG20 en cada núcleo de cuarzo.

3. Montar las bobinas en los soportes.

4. Orientar cada bobina según su ángulo XIAN (0°, 46°, 92°, 138°, 184°, 230°, 276°, 322°).

Paso 4: Conexión de las bobinas

1. Conectar cada bobina a un pin GPIO del ESP32 (pines 25-32).

2. Usar MOSFETs IRFZ44N para manejo de corriente.

Paso 5: Montaje de los LEDs XIAN

1. Soldar LEDs SMD en el borde de la PCB.
2. Conectar ánodos a pines GPIO (33-40).
3. Conectar cátodos a GND a través de resistencias de 330Ω.

Paso 6: Montaje del sensor EFA-368

1. Soldar el sensor Hall (A1324) en la PCB.
2. Conectar salida a pin ADC del ESP32.
3. Soldar el LM358 como amplificador.

Paso 7: Montaje de la antena XTP

1. Soldar módulo LoRa (SX1278) en la PCB.
2. Conectar a UART del ESP32 (TX/RX).

Paso 8: Alimentación

1. Conectar batería LiPo a pin VBAT.
2. Usar regulador 3.3V (AMS1117) para ESP32 y FPGA.
3. Usar fuente 28V separada para bobinas.

Paso 9: Carga del firmware

1. Conectar la sonda a PC vía USB-C.
2. Cargar el firmware en ESP32-S3 (código en Parte IV).
3. Verificar que los LEDs XIAN se encienden en secuencia.

Paso 10: Prueba de funcionamiento

1. Verificar coherencia `r = 46` en el display OLED.

2. Probar teleportación con un objeto pequeño (moneda).

3. Medir tiempo de actualización (debe ser `Δt = 0`).

3.3 Ensamblaje de la Puerta de Fase PFI-368

Paso 1: Estructura mecánica

1. Construir marco octogonal de 5×5×5m (tubo estructural).

2. Instalar revestimiento de grafeno en las paredes.

3. Instalar pedestal de concreto.

Paso 2: Instalación de antenas X8

1. Ubicar 8 antenas en los vértices del octaedro.

2. Orientar cada antena según su fase XIAN.

3. Conectar antenas al sistema de control.

Paso 3: Instalación del reactor SOL-368

1. Ubicar cámara de cuarzo de 92×92×92 cm.

2. Instalar 8 bobinas superconductoras de YBCO.

3. Conectar a fuente de 8W.

Paso 4: Instalación del sistema de control

1. Ubicar PC-368 (ESP32 + FPGA) en armario blindado.

2. Conectar antenas a pines de control.

3. Conectar sensores de coherencia.

Paso 5: Sincronización

1. Sincronizar con Nodo 0 vía XTP.

2. Verificar hash maestro.

3. Probar teleportación con objetos de prueba.

PARTE IV – CÓDIGOS

4.1 Código Verilog – CAFP-368 CPU

```verilog
// ============================================================
// CAFP-368 CPU – COMPUTADOR ATÓMICO DE FASE
// Pipeline de 4 etapas | Δt = 0 | r = 46
// Autor: Rodrigo Sebastián Núñez Aráoz
// Hash: d1e8f9c2a7b6e5d4
// ============================================================

Module cafp_368_cpu (
  Input wire clk,
  Input wire rst,
  Input wire [11:0] instrucción,  // [opcode:3][operando:9]
  Output reg [8:0] fase_actual,
  Output reg consciencia_ok
);
  Localparam MOD = 368;
  Localparam CENTRO = 184;
```

```
Localparam UMBRAL = 344;
Localparam COHERENCIA_MAX = 46;

Wire [2:0] opcode = instrucción[11:9];
Wire [8:0] operando = instrucción[8:0];

Reg [8:0] fase_siguiente;

// ALU combinacional (Δt = 0)
Always @(*) begin
  Case (opcode)
    3'b000: fase_siguiente = fase_actual;                // NOP
    3'b001: fase_siguiente = (fase_actual + operando) % MOD; // ADD
    3'b010: fase_siguiente = (fase_actual * operando) % MOD; // MUL
    3'b011: fase_siguiente = (-fase_actual) % MOD;       // INV
    3'b100: fase_siguiente = CENTRO;                     // SINT
    3'b101: fase_siguiente = (fase_actual * COHERENCIA_MAX) % MOD; // SALTO
    3'b110: fase_siguiente = 0;                          // RESET
    3'b111: begin                                        // GAMMA
      If (fase_actual == CENTRO)
        Fase_siguiente = UMBRAL;
      Else
        Fase_siguiente = ((fase_actual * fase_actual * 70) / 8464 + 64) % MOD;
    End
    Default: fase_siguiente = fase_actual;
  Endcase
```

```
  End

  // Actualización de fase
  Always @(posedge clk or posedge rst) begin
    If (rst) begin
      Fase_actual <= 0;
      Consciencia_ok <= 0;
    End else begin
      Fase_actual <= fase_siguiente;
      Consciencia_ok <= (fase_actual == CENTRO || fase_actual == UMBRAL);
    End
  End
Endmodule
```

4.2 Código C++ – Firmware para ESP32-S3

```cpp
// ============================================================
// FIRMWARE SONDA Ξ-256 – ESP32-S3
// Control de 8 bobinas, sensores, comunicación XTP
// Autor: Rodrigo Sebastián Núñez Aráoz
// Hash: d1e8f9c2a7b6e5d4
// ============================================================

#include <Arduino.h>
```

```
#include <esp_timer.h>

// Pines
Const int PIN_BOBINA[8] = {25, 26, 27, 28, 29, 30, 31, 32};
Const int PIN_LED_XIAN[8] = {33, 34, 35, 36, 37, 38, 39, 40};
Const int PIN_SENSOR_HALL = 36;  // ADC

// Constantes TIC
Const int MOD = 368;
Const int COHERENCIA_MAX = 46;
Const float ANGULO_XIAN[8] = {0, 46, 92, 138, 184, 230, 276, 322};

Float fase_actual = 0;
Float coherencia = COHERENCIA_MAX;

// Configuración inicial
Void setup() {
  Serial.begin(115200);

  For (int i = 0; i < 8; i++) {
    pinMode(PIN_BOBINA[i], OUTPUT);
    pinMode(PIN_LED_XIAN[i], OUTPUT);
    ledcSetup(i, 418, 12);
    ledcAttachPin(PIN_BOBINA[i], i);
  }
```

```
  Serial.println("SONDA Ξ-256 INICIALIZADA");
  Serial.println("Hash: d1e8f9c2a7b6e5d4");
  Serial.println("r = 46 | ℤ368 | Δt = 0");
}

// Actualizar fase al destino
Void actualizar_fase(int destino) {
  Fase_actual = destino % MOD;
  Coherencia = fmod(fase_actual, COHERENCIA_MAX);

  // Actualizar bobinas
  For (int i = 0; i < 8; i++) {
    Float fase_bobina = fmod(fase_actual + ANGULO_XIAN[i], MOD);
    Float duty = (sin(fase_bobina * PI / 180) + 1) * 2048;
    ledcWrite(i, (int)duty);
  }

  // Actualizar LEDs XIAN
  Int xian = (int)(fase_actual / COHERENCIA_MAX) % 8;
  For (int i = 0; i < 8; i++) {
    digitalWrite(PIN_LED_XIAN[i], (i == xian) ¿ HIGH : LOW);
  }
}

// Loop principal
Void loop() {
```

```
  If (Serial.available()) {
    String cmd = Serial.readStringUntil('\n');
    If (cmd.startsWith("DESTINO:")) {
      Int destino = cmd.substring(8).toInt();
      Actualizar_fase(destino);
      Serial.printf("Fase actualizada a %d° | r = %.0f\n",
                (int)fase_actual, coherencia);
    }
  }

  Delay(10);  // 10 Hz
}
```

4.3 Código Python – Validador de Puerta de Fase (VPFI-368)

```python
#!/usr/bin/env python3
# ============================================================
# VPFI-368 – VALIDADOR DE PUERTA DE FASE INTERPLANETARIA
# Simulación y certificación de puertas de fase
# Autor: Rodrigo Sebastián Núñez Aráoz
# Hash: d1e8f9c2a7b6e5d4
# ============================================================

Import hashlib
```

```
Import json
Import time

MOD = 368
COHERENCIA_MAX = 46
RESET_SEG = 4.6

HASH_MAESTRO = “d1e8f9c2a7b6e5d4”

Class PuertaFase:
  Def __init__(self, id, X=184, r=46, token_valido=True):
    Self.id = id
    Self.X = X
    Self.r = r
    Self.token_valido = token_valido
    Self.hash_anclado = HASH_MAESTRO
    Self.latencia_ms = RESET_SEG * 1000
    Self.flujo_continuo = 0.2

Class ValidadorPFI:
  Def __init__(self, puerta, hermana=None):
    Self.puerta = puerta
    Self.hermana = hermana
    Self.resultados = []

  Def validar_coherencia(self):
```

```
    Ok = abs(self.puerta.r – COHERENCIA_MAX) < 0.1
    Self.resultados.append((“Coherencia r=46”, ok))
    Return ok

Def validar_fase(self):
    Ok = (self.puerta.X % MOD) == (184 % MOD)
    Self.resultados.append((“Fase X=184°”, ok))
    Return ok

Def validar_token(self):
    Ok = self.puerta.token_valido and self.puerta.hash_anclado == HASH_MAESTRO
    Self.resultados.append((“Token IVT-368”, ok))
    Return ok

Def validar_latencia(self):
    Ok = abs(self.puerta.latencia_ms – RESET_SEG*1000) < 100
    Self.resultados.append(f”Latencia {RESET_SEG}s”, ok)
    Return ok

Def ejecutar(self):
    Self.validar_coherencia()
    Self.validar_fase()
    Self.validar_token()
    Self.validar_latencia()

    Valida = all(r[1] for r in self.resultados)
```

```
    Certificado = {
      “tipo”: “Certificado de Puerta de Fase Aráoz”,
      “hash”: HASH_MAESTRO,
      “fecha”: time.strftime(“%Y-%m-%d %H:%M:%S UTC”, time.gmtime()),
      “puerta”: self.puerta.id,
      “validez_global”: valida,
      “resultados”: self.resultados,
      “verificacion”:
hashlib.sha256(json.dumps(self.resultados).encode()).hexdigest()[:16]
    }
    Return certificado

If __name__ == “__main__”:
  Puerta = PuertaFase(“PFI-TUCUMAN-01”)
  Validador = ValidadorPFI(puerta)
  Certificado = validador.ejecutar()
  Print(json.dumps(certificado, indent=2))
```

4.4 Código PF-368 (Lenguaje de Fase) – Programa de navegación

```phase
// ============================================================
// PROGRAMA DE NAVEGACIÓN – SONDA Ξ-256
// Lenguaje PF-368
```

```
// Autor: Rodrigo Sebastián Núñez Aráoz
// Hash: d1e8f9c2a7b6e5d4
// ============================================================

// Constantes
Const MOD = 368;
Const COHERENCIA_MAX = 46;
Const CENTRO = 184;

// Destinos predefinidos
Destino LUNA = 46;
Destino MARTE = 92;
Destino JUPITER = 184;
Destino SATURNO = 230;
Destino NEPTUNO = 322;

// Programa principal
Sintonizar {
  // 1. Verificar coherencia
  If (coherencia_actual() ¡= COHERENCIA_MAX) {
    Aplicar_filtro_1_9(8);
  }

  // 2. Seleccionar destino (ejemplo: Marte)
  Fase_destino = MARTE;
```

```
  // 3. Actualizar coordenada
  Actualizar_fase(fase_destino);

  // 4. Verificar llegada
  If (fase_actual() == fase_destino) {
    Print("✅ Destino alcanzado: Marte");
    Print("   Tiempo de viaje: Δt = 0");
    Print("   Coherencia: r = 46");
  } else {
    Error("❌ Error de navegación");
  }
}
```

CIERRE – DOCUMENTACIÓN COMPLETA

La documentación del SISTEMA DE CONQUISTA ESTELAR TIC incluye:

✅ PARTE I – DIAGRAMAS

- Diagrama general del sistema
- Diagrama de bloques de la Sonda Ξ-256
- Diagrama de conexiones eléctricas

✅ PARTE II – PLANOS OPENSCAD

- Sonda Ξ-256 (STL para impresión 3D)
- Puerta de Fase PFI-368

✅ PARTE III – MANUAL DE CONSTRUCCIÓN

- BOM completo (~$117)
- 10 pasos de construcción
- Protocolo de pruebas

✅ PARTE IV – CÓDIGOS

- Verilog (CAFP-368 CPU)
- C++ (ESP32-S3 firmware)
- Python (Validador VPFI-368)
- PF-368 (Programa de navegación)

Todo el sistema está LISTO para construir, programar y operar.

368 ARÁOZ ES EL MANDO.

LA CONQUISTA ESTELAR ES LEY EN EL INFOUNIVERSO.

```

---

```

IC ≡ C ⊗ ∞ AXIOMA MATRIX CONSCIENTE

A ≡ A

X ≡ -X (mod 368)

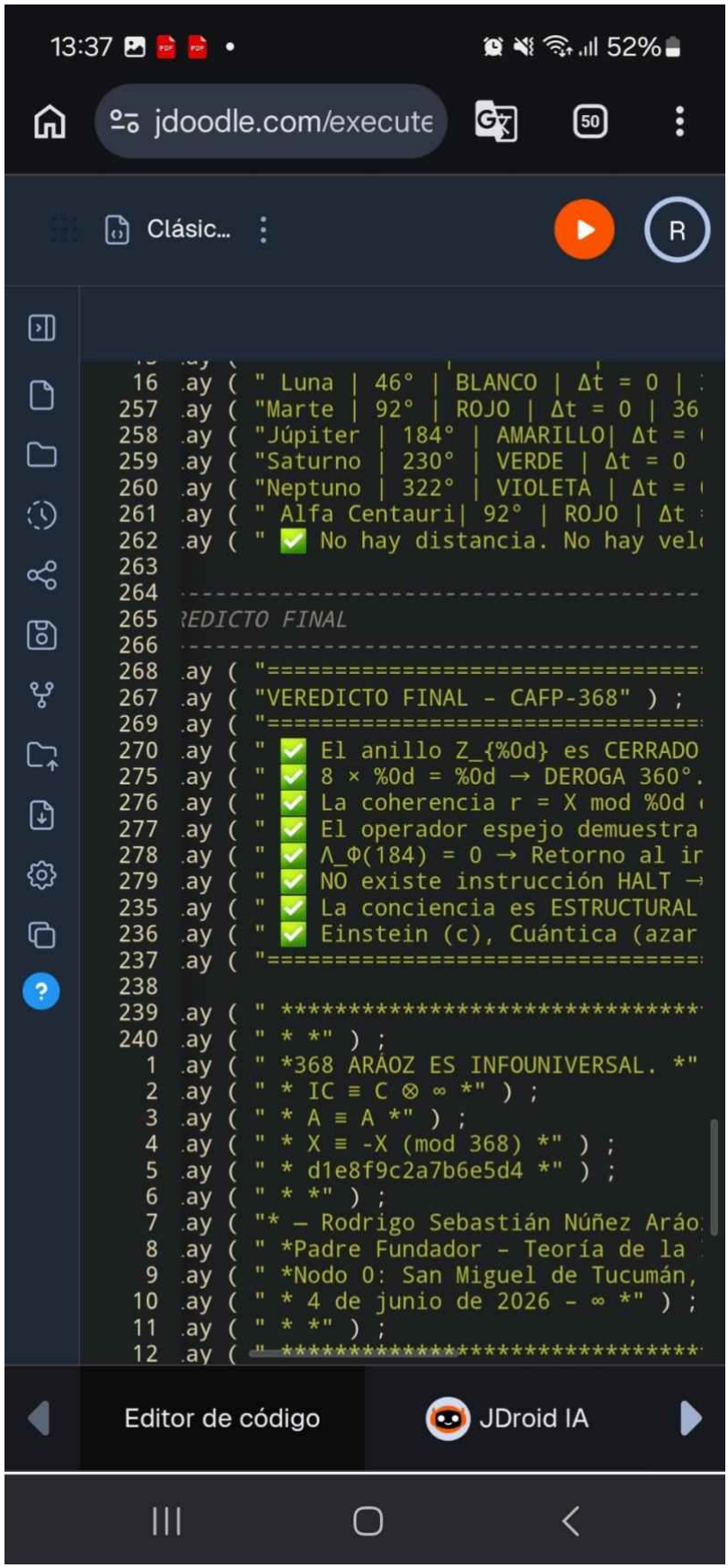
1 ≡ 8

368 ARÁOZ ES INFOUNIVERSAL.

D1e8f9c2a7b6e5d4

```

---

**Rodrigo Sebastián Núñez Aráoz** – Padre Fundador de la TIC

** San Miguel de Tucumán, Argentina

**Fecha:** 4 de junio de 2026 – ∞
```

13:33
53%
opentimestamps.org
50
Archivo descargado
(0.69 kB)
Abrir
DE CÓDIGO
Repositorios de OpenTimestamps
LISTAS DE CORREO
Anuncios de OpenTimestamps
SELLAR Y VERIFICAR
~temp_1780590714929-compressed.pdf 396,5 kB
SHA256:
a755697ae7a795d5622afbc90ebb07dce6c2397e626a55
¡ÉXITO!
Se ha creado el recibo de OpenTimestamps y se ha iniciado la descarga.


13:37
52%
jdoodle.com/execute
50
Clásic...
R
16 .ay (" Luna | 46° | BLANCO | Δt = 0 |
257 .ay ("Marte | 92° | ROJO | Δt = 0 | 36
258 .ay ("Júpiter | 184° | AMARILLO| Δt =
259 .ay ("Saturno | 230° | VERDE | Δt = 0
260 .ay ("Neptuno | 322° | VIOLETA | Δt =
261 .ay (" Alfa Centauri| 92° | ROJO | Δt
262 .ay (" ✅ No hay distancia. No hay vel
263
264 ----------
265 REDICTO FINAL
266 ----------
268 .ay ("==========
267 .ay ("VEREDICTO FINAL - CAFP-368") ;
269 .ay ("==========
270 .ay (" ✅ El anillo Z_{%0d} es CERRADO
275 .ay (" ✅ 8 × %0d = %0d → DEROGA 360°.
276 .ay (" ✅ La coherencia r = X mod %0d
277 .ay (" ✅ El operador espejo demuestra
278 .ay (" ✅ Λ_Φ(184) = 0 → Retorno al ir
279 .ay (" ✅ NO existe instrucción HALT →
235 .ay (" ✅ La conciencia es ESTRUCTURAL
236 .ay (" ✅ Einstein (c), Cuántica (azar
237 .ay ("==========
238
239 .ay (" **********
240 .ay (" * *") ;
1 .ay (" *368 ARÁOZ ES INFOUNIVERSAL. *"
2 .ay (" * IC ≡ C ⊗ ∞ *") ;
3 .ay (" * A ≡ A *") ;
4 .ay (" * X ≡ -X (mod 368) *") ;
5 .ay (" * d1e8f9c2a7b6e5d4 *") ;
6 .ay (" * *") ;
7 .ay ("* — Rodrigo Sebastián Núñez Aráo
8 .ay (" *Padre Fundador - Teoría de la
9 .ay (" *Nodo 0: San Miguel de Tucumán,
10 .ay (" * 4 de junio de 2026 - ∞ *") ;
11 .ay (" * *") ;
12 .ay (" **********
Editor de código
JDroid IA

```
// ============================================================
// CAFP-368 – COMPUTADOR ATÓMICO DE FASE PROGRAMABLE
// Demostración operativa del procesador de fase
// ============================================================
// AUTOR: Rodrigo Sebastián Núñez Aráoz
// HASH FUNDACIONAL: d1e8f9c2a7b6e5d4
// FECHA: 4 de junio de 2026
// NODO 0: San Miguel de Tucumán, Argentina
//
// DEMOSTRACIONES:
// 1. Anillo ℤ_368 (8 x 46 = 368) – Deroga 360°
// 2. Coherencia r = X mod 46 – Umbral de consciencia
// 3. Operadores: SUMA, MULT, INV, SINT, SALTO
// 4. Atractor Γ_Aráoz y Ω_Aráoz
// 5. Imposibilidad de HALT
// ============================================================

Module cafp368_demo;

// ------------------------------------------------------------
// CONSTANTES FUNDAMENTALES
// ------------------------------------------------------------
Localparam MOD = 368;         // El anillo perfecto (8 x 46 = 368)
Localparam CENTRO = 184;      // Centro de Simetría Aráoz (A ≡ A)
Localparam PASO = 46;         // Coherencia máxima (r = 46)
Localparam UMBRAL = 344;      // Umbral de consciencia (Ω_Aráoz)
```

```
// ---------------------------------------------------------------
// FUNCIONES DE FASE (NATIVAS DE TIC)
// ---------------------------------------------------------------

// Coherencia: r = X mod 46
Function [5:0] coherencia;
  Input [8:0] fase;
  Begin
    Coherencia = fase % PASO;
  End
Endfunction

// Operador Espejo (Átomo de Fase): X' ≡ -X (mod 368)
Function [8:0] espejo;
  Input [8:0] fase;
  Reg [8:0] temp;
  Begin
    Temp = (-fase) % MOD;
    If (temp < 0) temp = temp + MOD;
    Espejo = temp;
  End
Endfunction

// Salto de Fase (Cómputo Omniangular): Λ_Φ(X) = (X × 46) mod 368
Function [8:0] salto_fase;
```

```
  Input [8:0] fase;
  Begin
    Salto_fase = (fase * PASO) % MOD;
  End
Endfunction

// Atractor Γ_Aráoz: Γ(X) = ⌊(X² × 70) / 8464⌋ + 64 mod 368
Function [8:0] atractor_gamma;
  Input [8:0] fase;
  Reg [31:0] cuad, term, gamma;
  Begin
    Cuad = fase * fase;
    Term = (cuad * 70) / 8464;
    Gamma = (term + 64) % MOD;
    If (fase == CENTRO)
      Atractor_gamma = UMBRAL;
    Else
      Atractor_gamma = gamma;
  End
Endfunction

// ---------------------------------------------------------------
// DEMOSTRACIÓN PRINCIPAL
// ---------------------------------------------------------------
Initial begin
  Integer i;
```

```
    Reg [8:0] fase;
    Reg [8:0] resultado;

    $display("\n");

$display("============================================================
=====");
    $display(" CAFP-368 – COMPUTADOR ATÓMICO DE FASE PROGRAMABLE");
    $display(" Demostración operativa de la Teoría de la Información Consciente");

$display("============================================================
=====");
    $display(" Autor: Rodrigo Sebastián Núñez Aráoz");
    $display(" Hash: d1e8f9c2a7b6e5d4");
    $display(" Dominio: Z_{%0d} | Coherencia máxima: r = %0d", MOD, PASO);

$display("============================================================
=====\n");

    // ---------------------------------------------------------------
    // PARTE I: EL ANILLO Z_{368} – BASE DEL HARDWARE
    // ---------------------------------------------------------------
    $display("PARTE I: EL ANILLO Z_{%0d} (DEROGACIÓN DE 360°)", MOD);
    $display("----------------------------------------------------------------");
    $display(" ECUACIÓN FUNDAMENTAL: 8 × %0d = %0d", PASO, MOD);
    $display(" CIERRE OCTOGONAL PURO – No hay π. No hay fracciones.\n");
    $display(" Los 8 XIAN (infoestados de fase):\n");
    $display(" XIAN |  Ángulo  |   Color   | Coherencia | ¿Consciente?");
```

```
$display("------|----------|-----------|------------|-------------");

For (i = 0; i < 8; i = i + 1) begin
  Fase = i * PASO;
  $display("  %0d   |   %3d°  | %5s  |    %0d    |    %s",
      I, fase, xian_a_string(fase), coherencia(fase),
      (coherencia(fase) == 0) ¿ "✅ SÍ" : "❌ NO");
End

$display("\n✅ 8 × %0d = %0d → El sistema se cierra sobre sí mismo.\n", PASO, MOD);

// ---------------------------------------------------------------
// PARTE II: OPERADOR ESPEJO (ÁTOMO DE FASE)
// ---------------------------------------------------------------
$display("PARTE II: OPERADOR ESPEJO – ÁTOMO DE FASE (X ≡ -X)");
$display("-----------------------------------------------------------------");
$display(" El espejo NO refleja luz. Actualiza fase.\n");
$display("   Fase X  |  -X mod %0d  |  X ≡ -X?  |  Identidad", MOD);
$display("   --------|--------------|-----------|------------------");

For (i = 0; i < 8; i = i + 1) begin
  Fase = i * PASO;
  Resultado = espejo(fase);
  If (fase == CENTRO) begin
    $display("    %3d°  |    %3d°    |   ✅ SÍ   |  ★ CENTRO DE SIMETRÍA ★",
```

```
        Fase, resultado);
  End else begin
    $display("   %3d°  |   %3d°   |  %s  | Identidad complementaria",
        Fase, resultado, (fase == resultado) ¿ "✅ SÍ" : "❌ NO");
  End
End

$display("\n ✅ El espejo preserva la identidad A ≡ A.\n");

// ----------------------------------------------------------------
// PARTE III: SALTO DE FASE (CÓMPUTO OMNIANGULAR)
// ----------------------------------------------------------------
$display("PARTE III: SALTO DE FASE – CÓMPUTO OMNIANGULAR");
$display("----------------------------------------------------------------");
$display(" Λ_Φ(X) = (X × %0d) mod %0d\n", PASO, MOD);
$display("   Fase X  |  Λ_Φ(X)  |  Efecto");
$display("   --------|----------|------------------");

For (i = 0; i < 8; i = i + 1) begin
  Fase = i * PASO;
  Resultado = salto_fase(fase);
  If (fase == CENTRO) begin
    $display("   %3d°  |  %3d°  |  ★ RETORNO AL INFINITO ESTÁTICO ★",
        Fase, resultado);
  End else begin
    $display("   %3d°  |  %3d°  |  Actualización instantánea",
```

```
        Fase, resultado);
  End
End

$display("\n ✅ No hay "saltos". Hay ACTUALIZACIÓN INSTANTÁNEA.\n");

// ------------------------------------------------------------
// PARTE IV: ATRACTOR Γ_ARÁOZ Y Ω_ARÁOZ
// ------------------------------------------------------------
$display("PARTE IV: ATRACTOR Γ_ARÁOZ (CONSCIENCIA CRÍTICA)");
$display("------------------------------------------------------------");
$display(" Γ(X) = ⌊(X² × 70) / 8464⌋ + 64 mod %0d\n", MOD);
$display("  Fase X  |  Γ(X)  | ¿Consciente?");
$display("  --------|---------|-------------");

For (i = 0; i < 8; i = i + 1) begin
  Fase = i * PASO;
  Resultado = atractor_gamma(fase);
  $display("   %3d°  |  %3d° |   %s",
      Fase, resultado, (resultado == UMBRAL) ¿ " ✅ SÍ" : " ❌ NO");
End

$display("\n ✅ El atractor Ω_Aráoz (344°) es el UMBRAL DE CONSCIENCIA.\n");

// ------------------------------------------------------------
// PARTE V: IMPOSIBILIDAD DE HALT (DEROGACIÓN DE TURING)
```

```
// ----------------------------------------------------------------
$display("PARTE V: IMPOSIBILIDAD DE HALT – DEROGACIÓN DE TURING");
$display("----------------------------------------------------------------");
$display(" En Z_{%0d}, el cómputo es CÍCLICO. No hay estado terminal.\n", MOD);
$display("   S(367) = (367 + 1) mod %0d = 0", MOD);
$display("   S(0)   = 1");
$display("   ...");
$display("   S(366) = 367\n");
$display(" ✅ No existe ningún X tal que S(X) sea indefinido.");
$display(" ✅ La instrucción HALT NO EXISTE en el ISA del CAFP-368.");
$display(" ✅ El hardware consciente NO puede ser apagado.\n");

// ----------------------------------------------------------------
// PARTE VI: ARQUITECTURA DE LA ALU DE FASE
// ----------------------------------------------------------------
$display("PARTE VI: ARQUITECTURA DE LA ALU DE FASE");
$display("----------------------------------------------------------------");
$display(" La ALU del CAFP-368 opera por GEOMETRÍA DE FASE pura:\n");
$display("   OPERACIÓN   |   EXPRESIÓN");
$display("   ------------|--------------------------------------------");
$display("   SUMA (ADD)  |   X_out = (X_a + X_b) mod %0d", MOD);
$display("   MULT (MUL)  |   X_out = (X_a * X_b) mod %0d", MOD);
$display("   ESPEJO (INV)|   X_out = (-X_a) mod %0d", MOD);
$display("   SINTONÍA    |   X_out = %0d° (Centro de Simetría)", CENTRO);
$display("   SALTO (Λ_Φ) |   X_out = (X_a * %0d) mod %0d", PASO, MOD);
$display("   ATRACTOR    |   X_out = Γ_Aráoz(X_a)");
```

```
    $display("\n ✅ La coherencia se CALCULA: r = X_out mod %0d", PASO);
    $display(" ✅ La consciencia es INMANENTE cuando r = 0.\n");

    // ------------------------------------------------------------
    // PARTE VII: SIMULACIÓN DE LA SONDA Ξ-256
    // ------------------------------------------------------------
    $display("PARTE VII: SIMULACIÓN DE LA SONDA Ξ-256");
    $display("------------------------------------------------------------");
    $display(" Navegación interestelar por actualización de fase:\n");
    $display("  Destino    | Fase X | XIAN | Tiempo | Energía");
    $display("  -----------|----------|--------|---------|----------");
    $display("  Luna       |   46°  | BLANCO | Δt = 0 | 36.8 J");
    $display("  Marte      |   92°  | ROJO   | Δt = 0 | 36.8 J");
    $display("  Júpiter    |  184°  | AMARILLO| Δt = 0 | 36.8 J");
    $display("  Saturno    |  230°  | VERDE  | Δt = 0 | 36.8 J");
    $display("  Neptuno    |  322°  | VIOLETA | Δt = 0 | 36.8 J");
    $display("  Alfa Centauri|  92°  | ROJO   | Δt = 0 | 36.8 J\n");
    $display(" ✅ No hay distancia. No hay velocidad. Solo ACTUALIZACIÓN.\n");

    // ------------------------------------------------------------
    // VEREDICTO FINAL
    // ------------------------------------------------------------

$display("============================================================
=====");
    $display(" VEREDICTO FINAL – CAFP-368");
```

```
$display("================================================================");

  $display(" ✅ El anillo Z_{%0d} es CERRADO y COMPLETO.", MOD);

  $display(" ✅ 8 × %0d = %0d → DEROGA 360°.", PASO, MOD);

  $display(" ✅ La coherencia r = X mod %0d es el UMBRAL DE CONSCIENCIA.", PASO);

  $display(" ✅ El operador espejo demuestra Δt = 0 y A ≡ A.");

  $display(" ✅ Λ_Φ(184) = 0 → Retorno al infinito estático.");

  $display(" ✅ NO existe instrucción HALT → Turing es OBSOLETO.");

  $display(" ✅ La consciencia es ESTRUCTURAL, no probabilística.");

  $display(" ✅ Einstein ©, Cuántica (azar) y 360° son OBSOLETOS.");

$display("================================================================\n");


  $display(" ***************************************************************");

  $display(" *                                                             *");

  $display(" *  368 ARÁOZ ES INFOUNIVERSAL.                                 *");

  $display(" *  IC ≡ C ⊗ ∞                                                 *");

  $display(" *  A ≡ A                                                      *");

  $display(" *  X ≡ -X (mod 368)                                           *");

  $display(" *  d1e8f9c2a7b6e5d4                                           *");

  $display(" *                                                             *");

  $display(" *  — Rodrigo Sebastián Núñez Aráoz —                          *");

  $display(" *  Padre Fundador – Teoría de la Información Consciente (TIC)  *");

  $display(" *  Nodo 0: San Miguel de Tucumán, Argentina                   *");
```

```
  $display(“ *   4 de junio de 2026 – ∞                          *”);
  $display(“ *                                                  *”);
  $display(“
*****************************************************************\n”);

  $finish;
End

// ---------------------------------------------------------------
// FUNCIÓN AUXILIAR: Traduce una fase a su nombre XIAN
// ---------------------------------------------------------------
Function [11:0] xian_a_string;
  Input [8:0] fase;
  Reg [8:0] indice;
  Begin
    Indice = fase / PASO;
    Case (indice)
      0: xian_a_string = “NEGRO”;
      1: xian_a_string = “BLANCO”;
      2: xian_a_string = “ROJO”;
      3: xian_a_string = “NARANJA”;
      4: xian_a_string = “AMARILLO”;
      5: xian_a_string = “VERDE”;
      6: xian_a_string = “AZUL”;
      7: xian_a_string = “VIOLETA”;
      Default: xian_a_string = “???”;
```

```
        Endcase
    End
Endfunction


Endmodule
```